

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/EP04/053472

International filing date: 14 December 2004 (14.12.2004)

Document type: Certified copy of priority document

Document details: Country/Office: FR
Number: 0314959
Filing date: 19 December 2003 (19.12.2003)

Date of receipt at the International Bureau: 12 April 2005 (12.04.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)



World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse



22 FEB 2005

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le - 1 FEV. 2005

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT
NATIONAL DE
LA PROPRIÉTÉ
INDUSTRIELLE

SIEGE
26 bis, rue de Saint-Petersbourg
75800 PARIS cedex 08
Téléphone : 33 (0)1 53 04 53 04
Télécopie : 33 (0)1 53 04 45 23
www.inpi.fr





INSTITUT
NATIONAL DE
LA PROPRIÉTÉ
INDUSTRIELLE

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08

Téléphone : 33 (1) 53 04 53 04 Télécopie : 33 (1) 42 94 86 54

BREVET D'INVENTION CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI



N° 11354*03

REQUÊTE EN DÉLIVRANCE

page 1/2

BR1

Cet imprimé est à remplir lisiblement à l'encre noire

DB 540 0 W / 210502

REMISE DES PIÈCES DATE 19 DEC 2003 LIEU 54 INPI NANCY N° D'ENREGISTREMENT 0314959 NATIONAL ATTRIBUÉ PAR L'INPI DATE DE DÉPÔT ATTRIBUÉE 19 DEC. 2003 PAR L'INPI		1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE CABINET BALLOT 9, rue Claude Chappe Metz Technopôle 57070 METZ	
Vos références pour ce dossier (facultatif) 017069			
Confirmation d'un dépôt par télécopie		<input type="checkbox"/> N° attribué par l'INPI à la télécopie	
2 NATURE DE LA DEMANDE		Cochez l'une des 4 cases suivantes	
Demande de brevet		<input checked="" type="checkbox"/>	
Demande de certificat d'utilité		<input type="checkbox"/>	
Demande divisionnaire		<input type="checkbox"/>	
<i>Demande de brevet initiale</i> <i>ou demande de certificat d'utilité initiale</i>		N°	Date
		N°	Date
Transformation d'une demande de brevet européen <i>Demande de brevet initiale</i>		<input type="checkbox"/>	Date
		N°	Date
3 TITRE DE L'INVENTION (200 caractères ou espaces maximum) Procédé cryptographique d'exponentiation modulaire protégé contre les attaques de type DPA.			
4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE		Pays ou organisation Date <input type="text"/> N° Pays ou organisation Date <input type="text"/> N° Pays ou organisation Date <input type="text"/> N° <input type="checkbox"/> S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»	
5 DEMANDEUR (Cochez l'une des 2 cases)		<input checked="" type="checkbox"/> Personne morale <input type="checkbox"/> Personne physique	
Nom ou dénomination sociale		GEMPLUS	
Prénoms			
Forme juridique		Société Anonyme	
N° SIREN		<input type="text"/>	
Code APE-NAF		<input type="text"/>	
Domicile ou siège	Rue	Avenue du Pic de Bertagne Parc d'Activités de GEMENOS	
	Code postal et ville	13420 GEMENOS	
	Pays	FRANCE	
Nationalité		française	
N° de téléphone (facultatif)		N° de télécopie (facultatif)	
Adresse électronique (facultatif)			
<input type="checkbox"/> S'il y a plus d'un demandeur, cochez la case et utilisez l'imprimé «Suite»			

Remplir impérativement la 2^{ème} page



BREVET D'INVENTION CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE
page 2/2

BR2

REMISE DES PIÈCES DATE 19 DEC 2003 LIEU 54 INPI NANCY N° D'ENREGISTREMENT 0314959 NATIONAL ATTRIBUÉ PAR L'INPI	
---	--

DB 540 W / 210502

6 MANDATAIRE <i>(s'il y a lieu)</i>		
Nom		LECLAIRE
Prénom		Jean-Louis
Cabinet ou Société		CABINET BALLOT
N° de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	9, rue Claude Chappe Metz Technopôle
	Code postal et ville	57 10 17 10 METZ
	Pays	FRANCE
N° de téléphone <i>(facultatif)</i>		03.87.74.81.36
N° de télécopie <i>(facultatif)</i>		03.87.36.26.76
Adresse électronique <i>(facultatif)</i>		
7 INVENTEUR (S)		Les inventeurs sont nécessairement des personnes physiques
Les demandeurs et les inventeurs sont les mêmes personnes		<input type="checkbox"/> Oui <input checked="" type="checkbox"/> Non : Dans ce cas remplir le formulaire de Désignation d'inventeur(s)
8 RAPPORT DE RECHERCHE		Uniquement pour une demande de brevet (y compris division et transformation)
Établissement immédiat ou établissement différé		<input checked="" type="checkbox"/> <input type="checkbox"/>
Paiement échelonné de la redevance <i>(en deux versements)</i>		Uniquement pour les personnes physiques effectuant elles-mêmes leur propre dépôt <input type="checkbox"/> Oui <input checked="" type="checkbox"/> Non
9 RÉDUCTION DU TAUX DES REDEVANCES		Uniquement pour les personnes physiques <input type="checkbox"/> Requête pour la première fois pour cette invention <i>(joindre un avis de non-imposition)</i> <input type="checkbox"/> Obtenue antérieurement à ce dépôt pour cette invention <i>(joindre une copie de la décision d'admission à l'assistance gratuite ou indiquer sa référence)</i> : AG [] [] [] [] []
10 SÉQUENCES DE NUCLEOTIDES ET/OU D'ACIDES AMINÉS		<input type="checkbox"/> Cochez la case si la description contient une liste de séquences
Le support électronique de données est joint		<input type="checkbox"/>
La déclaration de conformité de la liste de séquences sur support papier avec le support électronique de données est jointe		<input type="checkbox"/>
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes		
11 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire) Jean-Louis LECLAIRE - 93.4009		VISA DE LA PRÉFECTURE OU DE L'INPI CABINET BALLOT CONSEILS EN PROPRIÉTÉ INDUSTRIELLE 9, rue Claude Chappe Technopôle Metz 2000 57070 METZ Magali DEMANGE

PROCEDE CRYPTOGRAPHIQUE D'EXPONENTIATION MODULAIRE PROTEGE CONTRE
LES ATTAQUES DE TYPE DPA

Dans le domaine de la protection des algorithmes cryptographiques contre les attaques DPA, l'invention concerne un procédé au cours duquel on réalise une exponentiation modulaire de type x^d , avec d un exposant entier de $m+1$ bits, en balayant les bits de d de gauche à droite dans une boucle indicée par i variant de m à 0 et en calculant et en mémorisant dans un accumulateur (R0), à chaque tour de rang i , un résultat partiel actualisé égal à $x^{b(i)}$. $b(i)$ correspond aux $m-i+1$ bits de poids les plus forts de l'exposant d : $b(i) = d_{m \rightarrow i}$. Le nombre constitué des bits de poids j à k de d est défini par :

$$d_{k \rightarrow j} = (d_k, \dots, d_j)_2 = \sum_{i=j}^k d_i \cdot 2^{(i-j)}.$$

L'exponentiation modulaire est une des opérations élémentaires utilisés dans de nombreux cryptosystèmes, tels que les cryptosystèmes RSA (Rivest, Shamir and Adleman) ou les cryptosystèmes DH (Diffie and Hellman). Pour de telles applications, x est par exemple un message à chiffrer ou déchiffrer, à signer ou à authentifier, et d est par exemple une clé publique, une clé secrète, ou une partie d'une telle clé.

Depuis l'invention de la cryptographie à clé publique par Diffie et Hellman, de nombreux cryptosystèmes à clé publique ont été proposés. Parmi ceux qui résistent à l'analyse cryptographique, le cryptosystème RSA est sans aucun doute le plus largement utilisé. Sa sécurité intrinsèque réside dans la difficulté de factoriser des grands nombres entiers. En dépit d'intensives recherches, le problème de la factorisation est encore considéré comme un important problème, rendant le cryptosystème RSA

sûr pour des applications sensibles comme par exemple le chiffrement de données ou la signature digitale.

Aussi, plutôt que de tenter de casser l'algorithme RSA à un niveau mathématique, les cryptographes se sont
5 intéressés aux implémentations concrètes des cryptosystèmes RSA. Ceci a conduit à l'essor des attaques par fautes et des attaques à canaux cachés, visant à découvrir notamment des informations confidentielles (comme par exemple des clés ou des parties des clés)
10 manipulées au cours de l'une ou de l'autre des étapes mises en œuvre par le dispositif de calcul exécutant une opération cryptographique.

Les attaques à canaux cachés les plus connues sont dites simples ou différentielles. On entend par attaque à canal
15 caché simple (SPA) ou différentielle (DPA), une attaque basée sur la mesure d'une grandeur physique depuis l'extérieur du dispositif, dont l'analyse directe (attaque simple SPA) ou l'analyse selon une méthode statistique (attaque différentielle DPA) permet de
20 découvrir des informations manipulées dans le dispositif. Ces attaques ont notamment été dévoilées par Paul Kocher (Advances in Cryptology - CRYPTO'99, vol. 1666 of Lecture Notes in Computer Science, pp.388-397. Springer-Verlag, 1999).

25 Parmi les grandeurs physiques qui peuvent être exploitées à ces fins, on peut citer le temps d'exécution, la consommation en courant, le champ électromagnétique rayonné par la partie du composant utilisée pour exécuter le calcul, etc. Ces attaques sont basées sur le fait que,
30 au cours de l'exécution d'un algorithme, la manipulation d'un bit, c'est à dire son utilisation par une instruction particulière, laisse une empreinte

particulière sur la grandeur physique considérée, selon la valeur de ce bit et / ou selon l'instruction.

Il existe deux familles d'implémentations des algorithmes d'exponentiation permettant d'évaluer la valeur de $y = x^d \bmod N$: les implémentations dites de droite à gauche et les implémentations dites de gauche à droite.

Dans les implémentations de gauche à droite, on balaye les bits de l'exposant depuis le bit de poids le plus fort jusqu'au bit de poids le plus faible. Dans cette deuxième famille d'algorithmes d'exponentiation, est notamment connu l'algorithme SAM (pour Square And Multiply ou élever au carré et multiplier) et ses variantes telles que les algorithmes à fenêtre glissante. Par rapport aux algorithmes dits de droite à gauche, les algorithmes de gauche à droite nécessitent moins de mémoire et permettent l'utilisation de puissances x^i précalculées pour accélérer le calcul de y . Tous les algorithmes de gauche à droite ont en commun l'utilisation d'un accumulateur (ou registre) qu'on actualise tout au long du calcul pour mémoriser la valeur de $x^{d_{m \rightarrow i}} \bmod N$ pour des valeurs décroissantes de i jusqu'à ce que l'accumulateur contienne la valeur finale $y = x^{d_{m \rightarrow 0}} = x^d \bmod N$. $d_{k \rightarrow j}$ est le mot constitué des bits de poids j à k de d .

Le principe général de l'algorithme SAM est le suivant. On note $d = (d_m, \dots, d_0)_2 = \sum_{i=0}^m d_i \cdot 2^i$, la représentation binaire de l'exposant d , avec $d_i \in \{0, 1\}$, le bit de poids i de d . Pour chaque bit de d , l'algorithme SAM mémorise dans un accumulateur (registre R0) un résultat actualisé calculé à partir de la relation de récurrence

$x^{d_{m \rightarrow i}} = (x^{d_{m \rightarrow i+1}})^2 * x^{d_i}$, avec $x^{d_{m \rightarrow m}} = x^{d_m}$, ce qui est résumé par l'algorithme ci-dessous :

```

    Entrée : x, d = (d_m, ..., d_0)_2
    Sortie : y = x^d mod N
5      R0 <- 1 ; R2 <- x, i <- m
      tant que i ≥ 0, faire :
        R0 <- R0×R0 mod N
        si d_i = 1 alors R0 <- R0×R2 mod N
        i <- i-1
10     fin tant que
        retourner R0

```

R0 <- x signifie que l'on mémorise dans le registre R0 la valeur de x. R0×R0 signifie que l'on réalise une mise au carré du contenu du registre R0. R0×R2 signifie qu'on
 15 réalise le produit du contenu du registre R0 par le contenu du registre R2. Enfin, $d_{i \rightarrow j}$ fait référence aux bits de rang j à i de d.

20 Pour prévenir les attaques d'implémentation, il est connu qu'il faut rendre les algorithmes aléatoires. Dans le cas du cryptosystème RSA, on connaît actuellement deux types de contre-mesures pour rendre aléatoire le calcul de $y = x^d \bmod N$.

25 Le premier type de contre-mesure consiste à rendre aléatoire les données d'entrée de l'algorithme.

Un premier exemple de cette première contre-mesure consiste à rendre aléatoire la donnée x avant de réaliser l'exponentiation modulaire, en ajoutant à x un terme aléatoire et à faire les calculs modulo $2^k N$, avant un
 30 modulo N final :

$\bar{x} \leftarrow x + r1.N$, avec r1 un nombre aléatoire de k-bits

et à faire les calculs modulo $(2^k).N$, avant une réduction modulo N finale. Cette première contre-mesure, décrite par P. Kocher, présente l'avantage d'être indépendante de l'algorithme d'exponentiation.

- 5 Un deuxième exemple de cette première contre-mesure consiste à rendre aléatoire l'exposant d avant de réaliser l'exponentiation modulaire, en lui ajoutant un terme aléatoire :

$$\bar{d} \leftarrow d + r2.\phi(N), \text{ } r2 \text{ un nombre aléatoire de } k\text{-bits.}$$

- 10 Le plus souvent, ces deux solutions sont combinées pour réaliser l'opération $y = \bar{y} \bmod N$ avec $\bar{y} = \bar{x}^{\bar{d}} \bmod (2^k.N)$.

- 15 Un troisième exemple de cette première contre-mesure utilisée seule par exemple lorsque x est le résultat d'un formatage probabiliste (par exemple à l'aide de la fonction PSS ou Probabilistic Signature Scheme), car dans ce cas, x est déjà masqué et on calcule directement $y = x^{\bar{d}} \bmod N$ avec $\bar{d} = d + r2.\phi(N)$ avec $r2$ aléatoire.

- 20 Malheureusement, une telle randomisation de l'exposant d est limitée à des implémentations particulières, appelées implémentations CRT, du cryptosystème RSA car la valeur de la constante d'Euler $\phi(N)$ n'est généralement pas connue de l'algorithme d'exponentiation privé dans sa version standard (c'est-à-dire non CRT).

- 25 La deuxième contre-mesure consiste à rendre l'algorithme d'exponentiation lui-même aléatoire. La meilleure mise en pratique de la 2^{ème} contre-mesure est l'algorithme MIST de Walter. L'algorithme MIST génère de manière aléatoire une nouvelle chaîne d'addition pour l'exposant d pour
30 réaliser $x^d \bmod N$. Pour minimiser le nombre de registres, la chaîne d'addition est réalisée à la volée

par l'intermédiaire d'une adaptation d'un algorithme d'exponentiation basé sur des chaînes de divisions. Un autre exemple est une version améliorée d'un algorithme à fenêtre glissante (voir Kouichi Itoh, Jun Yajima, Masahiko Takenaka and Naoya Torii. DPA countermeasures by improving the window method CHES 2002, volume 2523 of Lecture Notes in computer Science, pages 303-317, Springer Verlag 2002). Comparé à la première contre-mesure, ceci permet de rendre aléatoire l'exponentiation sans avoir besoin de connaître $\phi(N)$ mais nécessite un algorithme de division sécurisé pour calculer les chaînes de divisions et cause des soucis de gestion des calculs non négligeables.

L'invention propose un nouveau procédé pour rendre aléatoire l'exécution d'une exponentiation modulaire, dans le but de prévenir les attaques différentielles (DPA), présentant les avantages des deux contre-mesures connues : comme dans la première contre-mesure, le procédé selon l'invention n'impose pas d'algorithme d'exponentiation particulier et s'applique à tout algorithme d'exponentiation, et comme dans la deuxième contre-mesure, dans l'invention, l'algorithme lui-même est rendu aléatoire, et non plus seulement les données qu'il manipule. Ainsi, l'algorithme n'a pas besoin de connaître $\phi(N)$ et / ou la clé publique e dans une exponentiation RSA (la clé e est souvent indisponible à l'algorithme de signature ou de déchiffrement).

Le procédé selon l'invention introduit le concept de exponentiation auto-aléatoire, signifiant que l'exposant d est utilisé lui-même comme une source additionnelle d'aléatoire dans le processus d'exponentiation.

Ainsi, l'invention concerne un procédé cryptographique au cours duquel on réalise une exponentiation modulaire de

type x^d , avec d un exposant entier de $m+1$ bits, en balayant les bits de d de gauche à droite dans une boucle indicée par i décrémenté de m à 0 par pas de 1 et en calculant et en mémorisant dans un accumulateur, à chaque

5 tour de rang i , un résultat partiel actualisé égal à $x^{b(i)}$, $b(i)$ étant les $m-i+1$ bits de poids les plus forts de l'exposant d .

Le procédé selon l'invention est caractérisé en ce que :

- à la fin d'un tour de rang $i(j)$ ($i = i(0)$) choisi

10 aléatoirement, on réalise une étape E1 de randomisation au cours de laquelle :

E1 : on soustrait un nombre z ($z = b(i(j))$, $z = b(i(j)).2^r$, $z = u$) aléatoire à une partie des bits de d non encore utilisés ($d_{i-1} \rightarrow 0$) dans le procédé

15 - puis, après avoir utilisé les bits de d modifiés par l'étape de randomisation E1, on réalise une étape de consolidation E2 au cours de laquelle :

E2 : on mémorise ($R0 \leftarrow R1 \times R0$) dans l'accumulateur ($R0$) le résultat de la multiplication du contenu de

20 l'accumulateur ($x^{b(i)}$) par un nombre fonction de x^z mémorisé dans un registre ($R1$).

D'un point de vue pratique, au cours de l'étape E1, le nombre z est soustrait au contenu d'un registre dans lequel est mémorisé initialement l'exposant d , et le

25 résultat de la soustraction est mémorisé dans le même registre, puis on continue de balayer les bits de b .

Pour que le résultat de l'exponentiation $x^d \bmod N$ soit correct à la fin du procédé, l'étape de randomisation E1, ne doit pas modifier les bits de d déjà utilisés dans le

30 calcul (on rappelle que le procédé utilise un algorithme de gauche à droite). L'index $i(j)$ auquel on réalise la

randomisation E1, choisi aléatoirement, doit donc être choisi tel que les $m-i(j)+1$ bits de poids les plus forts du registre contenant initialement l'exposant d restent inchangés lors de l'étape E1. On appellera par la suite
5 cette condition une condition de "consistance".

L'idée essentielle de l'invention est ainsi d'utiliser un découpage du calcul de $x^d \bmod N$ de la forme : $x^d = x^{(d-z)} * x^z$ (décrit dans la demande de brevet français n°
10 02 04117 (n° à confirmer) avec z un nombre aléatoire utilisé comme moyen de masquage de l'exposant d. On choisit de préférence des valeurs de z appropriées telles que x^z puisse être obtenu facilement à partir de x^b déjà calculé par ailleurs au cours du procédé. A noter
15 qu'un choix totalement aléatoire de z entraîne un quasi doublement du temps de calcul.

Le procédé selon l'invention s'applique indépendamment de l'algorithme d'exponentiation de gauche à droite. Par ailleurs, le rang $i(j)$ auquel on réalise l'étape E1 est
20 choisi aléatoire donc le procédé lui-même est aléatoire, et non pas seulement les données qu'il manipule.

Le procédé selon l'invention est également efficace en termes de place (il ne nécessite qu'un seul registre de calcul supplémentaire) et en terme de temps de calcul,
25 comme on le verra mieux par la suite dans l'exemple de l'algorithme SAM.

Le procédé selon l'invention est encore facile à mettre en œuvre quel que soit l'algorithme auquel il est appliqué : il ne repose sur aucune propriété de groupe et
30 sa mise en œuvre ne nécessite pas de connaître au

préalable l'ordre du groupe dans lequel l'exponentiation est réalisée.

Enfin, le procédé selon l'invention peut être utilisé conjointement avec d'autres mesures de protection des algorithmes, comme par exemple les contre mesures dévoilées par P. Kocher et rappelées précédemment.

L'étape E1 de randomisation peut être réalisée une seule fois au cours du procédé. L'étape E1 peut également être réalisée plusieurs fois, à la fin de différents tours de rang $i(j)$ (c'est-à-dire au rang $i = i(0)$, puis au rang $i = i(1)$, ..., puis enfin au rang $i = i(f)$) choisis aléatoirement entre 0 et m . L'idée est ici d'améliorer encore la sécurité du procédé en utilisant la relation :

$$\begin{aligned} x^d &= x^{(d-z_1-z_2-\dots-z_f)} x^{z_1} x^{z_2} \times \dots \times x^{z_f} \\ &= x^{(((d-z_1)-z_2)-\dots-z_f) \times ((x^{z_1}) x^{z_2}) \times \dots \times x^{z_f}} \end{aligned}$$

On peut choisir au début du procédé le ou les rangs $i(j)$ aléatoires auxquels on réalise une randomisation E1. Par exemple, au début du procédé, on détermine un ensemble prédéfini $\{i(0), i(1), \dots, i(f)\}$ de $f+1$ (f étant aléatoire ou non) valeurs de l'index i pour lesquelles on souhaite réaliser une randomisation E1. Dans ce cas, à chaque tour, on décide de réaliser ou non une randomisation E1 selon que l'index courant i fait partie ou non de l'ensemble prédéfini.

On peut aussi choisir aléatoirement au début de chaque tour i de réaliser ou non l'étape E1 de randomisation. Dans ce cas, on utilise par exemple une variable booléenne p , tirée aléatoirement à la fin de chaque tour d'index i .

Différents modes de réalisation de l'invention vont maintenant être décrits, qui diffèrent l'un de l'autre essentiellement par le mode de réalisation de l'étape E1, et notamment par le choix de z et par le choix de la partie de d à laquelle on soustrait z .

Selon un premier mode de réalisation, on choisit de réaliser l'étape de consolidation une seule fois à la fin du procédé. Ceci impose de soustraire z systématiquement aux bits de poids les plus faibles de l'exposant d , de sorte à obtenir un résultat correct à la fin du procédé.

Selon une première variante de ce mode de réalisation, on choisit $z = b(i(j)) = d_{m \rightarrow i(j)}$ pour un nombre aléatoire $i(j)$ choisi et, au cours de l'étape de randomisation E1, on soustrait $b(i(j))$ à d , c'est-à-dire aux bits de poids les plus faibles de d .

Le choix $z = b(i(j))$ est particulièrement avantageux puisque $x^{b(i(j))} = x^{d_{m \rightarrow i(j)}}$ est déjà disponible dans l'accumulateur à la fin du tour $i(j)$ et n'a donc pas besoin d'être calculé. La variable $i(j)$ est choisie telle que les bits de poids $i(j)$ à m du nombre $d - b(i(j))$ soient égaux aux bits de poids $i(j)$ du nombre d , de sorte que les $m - i(j) + 1$ premiers tours du calcul de x^d sont identiques aux $m - i(j) + 1$ premiers tours du calcul de $x^{(d - b(i(j)))}$ (condition de consistance). A la fin du tour $i(j)$, on calcule $d - z = d - b(i(j))$ et on mémorise le contenu de l'accumulateur x^b dans le registre (E1).

Dans un exemple particulier, une variable booléenne p est utilisée pour déterminer, à la fin de chaque tour d'index i , si on réalise ou non une randomisation. Si p prend une valeur active, alors on réalise l'étape E1 : on remplace
 5 le nombre d par le nombre $d - b(i(j))$ et on mémorise $x^{b(i(j))}$.

Comme dans l'algorithme de gauche à droite classique, l'accumulateur $R0$ est utilisé pour conserver la valeur de $x^{d_{m \rightarrow i}}$, à chaque tour d'index i . On utilise le registre
 10 $R1$ pour conserver le produit : $\prod_j x^{d_{m \rightarrow i(j)}}$.

Le tout appliqué à l'algorithme SAM connu, on obtient le l'algorithme I suivant :

```

    Entrée :  $x, d = (d_m, \dots, d_0)_2$ 
    Sortie :  $y = x^d \bmod N$ 
15       $R0 \leftarrow 1 ; R1 \leftarrow 1 ; R2 \leftarrow x ; i \leftarrow m$ 
      tant que  $i \geq 0$ , faire :
         $R0 \leftarrow R0 \times R0 \bmod N$ 
        si  $d_i = 1$  alors  $R0 \leftarrow R0 \times R2 \bmod N$ 
         $p \leftarrow R\{0, 1\}$ 
20      si  $((p = 1) \text{ ET } d_{i-1 \rightarrow 0} \geq d_{m \rightarrow i})$  alors
           $d \leftarrow d - d_{m \rightarrow i}$ 
           $R1 \leftarrow R1 \times R0 \bmod N$ 
        fin si
         $i \leftarrow i - 1$ 
25      fin tant que
       $R0 \leftarrow R0 \times R1 \bmod N$ 
    retourner  $R0$ 
```

$p \leftarrow R\{0, 1\}$ signifie que l'on choisit la valeur de p aléatoirement dans l'ensemble $\{0, 1\}$. p est ainsi une
 30 variable booléenne aléatoire.

L'étape de randomisation E1 ($d \leftarrow d - d_{m \rightarrow i(j)} ; R1 \leftarrow R1 \times R0 \bmod N$) est réalisée uniquement si $p = 1$ (c'est-à-

dire si une randomisation doit être réalisée) et si $d_{i(j)-1 \rightarrow 0} \geq d_{m \rightarrow i(j)}$.

La condition $d_{i(j)-1 \rightarrow 0} \geq d_{m \rightarrow i(j)}$ signifie que les bits de poids 0 à $i-1$ de d sont supérieurs à $b(i(j))$, $b(i(j))$ étant égal aux bits de poids $i(j)$ à m de d . Ceci permet de garantir que les $m-i+1$ bits de poids les plus forts de $d-b(i(j))$ sont identiques aux $m-i+1$ bits de poids forts de d , et donc que les $m-i+1$ premiers tours du calcul de x^d sont identiques aux $m-i+1$ premiers tours du calcul de $x^{(d-b(i(j)))}$.

La condition de "consistance" ($d_{i(j)-1 \rightarrow 0} \geq d_{m \rightarrow i(j)}$) implique que seulement les bits de poids faibles de l'exposant d sont rendus aléatoires. De plus, on remarque que l'étape de randomisation $d \leftarrow d - d_{m \rightarrow i(j)}$ modifie uniquement les $(m-i(j)+1)$ bits les moins significatifs de d .

A noter que, dans l'algorithme I, comme à l'itération $i = i(j)$ l'étape de mise à jour $d \leftarrow d - d_{m \rightarrow i}$ ne modifie pas les $(m-i+1)$ bits les plus significatifs de d , cette étape peut être remplacée par l'étape équivalente :

$$d_{i-1 \rightarrow 0} \leftarrow d_{i-1 \rightarrow 0} - d_{m \rightarrow i}.$$

Selon une deuxième variante du premier mode de réalisation, on choisit z égal $g.b(i)$, avec g un nombre aléatoire tel que $d_{i(j)-1 \rightarrow 0} \geq g.d_{m \rightarrow i(j)}$. On utilise dans ce cas la relation $x^d = x^{(d-z)} \cdot x^z = x^{(d-g.b(i))} \cdot (x^{b(i)})^g$. et, d'un point de vue pratique, pour réaliser une randomisation E1 à la fin du tour d'index $i(j)$:

- on calcule $z = g.b(i)$ et on soustrait le résultat à l'exposant d ,

- le registre R1 est mis à jour en multipliant son contenu par le contenu de l'accumulateur $(x^b(i))$ exposé à la puissance g . Ce qui concrètement peut être réalisé par l'instruction $R1 \leftarrow R1 \times R0^g \bmod N$.

5 On choisit de préférence $g = 2^\tau$, τ étant un nombre entier aléatoire. Ceci simplifie considérablement les calculs car le calcul de $g.b(i) = g.d_{m \rightarrow i(j)}$ revient à un simple décalage de bits et l'évaluation de $(x^b(i))^g \bmod N$ revient à réaliser le calcul de τ carrés.

10 Puisque multiplier par 2^τ revient à un décalage de bits, l'instruction $d \leftarrow d - 2^\tau.d_{m \rightarrow i}$ qui calcule $d - g.b(i)$ peut être remplacée par $d_{m \rightarrow \tau} \leftarrow d_{m \rightarrow \tau} - d_{m \rightarrow i}$ ou mieux, par l'instruction équivalente $d_{i-1 \rightarrow \tau} \leftarrow d_{i-1 \rightarrow \tau} - d_{m \rightarrow i}$.

De plus, comme pour les autres modes de réalisation, on
15 doit vérifier que à l'itération $i=i(j)$, $d_{i-1 \rightarrow 0} \geq 2^\tau.d_{m \rightarrow i}$. Cette condition de consistance peut être remplacée par un test équivalent mais plus efficace : $d_{i-1 \rightarrow \tau} \geq d_{m \rightarrow i}$.

De préférence, on choisit τ aléatoire dans l'ensemble $\{0, \dots, T\}$. La borne T est choisie comme le meilleur compromis
20 entre la randomisation des bits les plus significatifs de d et l'efficacité (en terme de temps de calcul notamment) du calcul des τ carrés.

Dans l'exemple particulier de l'algorithme SAM, on obtient finalement l'algorithme I' suivant.

Entrée : $x, d = (d_m, \dots, d_0)_2$

Sortie : $y = x^d \bmod N$

```

5      R0 <- 1 ; R1 <- 1 ; R2 <- x ; i <- m
      tant que i ≥ 0, faire :
          R0 <- R0×R0 mod N
          si  $d_i = 1$  alors R0 <- R0×R2 mod N
           $\rho \leftarrow R\{0, 1\}$  ;  $\tau \leftarrow R\{0, \dots, T\}$ 
10      si (( $\rho = 1$ ) ET ( $d_{i-1 \rightarrow \tau} \geq d_{m-i}$ )) alors
           $d_{i-1 \rightarrow \tau} \leftarrow d_{i-1 \rightarrow \tau} - d_{m-i}$ 
          R3 <- R0
          tant que ( $\tau > 0$ ) faire
              R3 <-  $R3^2 \bmod N$  ;  $\tau \leftarrow \tau - 1$ 
15      fin tant que
          R1 <- R1×R3 mod N
          fin si
          i <- i-1
      fin tant que
20      R0 <- R0×R1 mod N
      retourner R0
  
```

Un avantage de l'algorithme I' est qu'on randomise en partie la moitié supérieure de d et qu'on ajoute en conséquence de l'entropie (c'est-à-dire de l'aléatoireité). Par contre, un registre supplémentaire R3 est nécessaire pour calculer $R0^{2^\tau}$.

Les algorithmes I et I' peuvent être suffisants pour protéger les exposants dans certains cas. Par exemple, en raison de sa construction, le cryptosystème RSA dévoile toujours la moitié la plus significative de l'exposant d privé si l'exposant public correspondant est petit. Rendre aléatoire les bits de poids forts de d

n'apporterait donc aucune protection pour un tel algorithme.

Toutefois, pour d'autres algorithmes et dans d'autres situations, rendre aléatoire tous les bits de d apporterait une sécurité supplémentaire.

Pour cela, on propose, dans un deuxième mode de réalisation, de choisir $z = b(i(j)) = d_{m \rightarrow i(j)}$ pour un nombre aléatoire $i(j)$ et, au cours de l'étape E1, on soustrait $b(i)$ non plus à d , mais à une partie des bits de d correspondant aux bits de d de poids $i(j) - c(j)$ à $i(j) - 1$, $c(j)$ étant un nombre entier tel que $i(j) \geq c(j) \geq 0$. Ceci peut être exprimé par l'instruction suivante :

$$d_{m \rightarrow i(j) - c(j)} \leftarrow d_{m \rightarrow i(j) - c(j)} - d_{m \rightarrow i(j)}$$

Préférentiellement, comme lors de l'étape E1 d'une randomisation au rang $i(j)$, on modifie les bits de poids $i(j) - c(j)$ à $i(j) - 1$ de d et on choisit de ne réaliser qu'une seule randomisation à la fois, et on choisit de réaliser une étape de consolidation à la fin du rang utilisant le dernier bit de d modifié lors de l'étape de randomisation E1 précédente (et non plus à la fin du procédé), c'est à dire après l'évaluation du résultat partiel $x^{(d_{m \rightarrow i(j) - c(j)})} \bmod N$.

Ceci revient à imposer la condition $i(j+1) \leq i(j) - c(j)$, $i(j+1)$ étant l'index de la randomisation suivante. Ceci permet de ne pas utiliser de registres supplémentaires pour mémoriser les bits de l'exposant qui ont été modifiés lors d'une randomisation précédente. Egalement, on choisit $i(j) - c(j) < 0$, de sorte que $i(j) - c(j)$ puisse être utilisé pour définir le rang d'un bit de d pour le calcul de $x^{(d_{m \rightarrow i(j) - c(j)})} \bmod N$.

Ces deux conditions peuvent être concrétisées par l'utilisation d'un sémaphore booléen σ qui indique si une mise à jour est autorisée ou pas : σ a une valeur inactive tant que $i \geq i(j) - c(j)$ et est activée lorsque
 5 $i < i(j) - c(j)$. Elle devient de plus inutilisable dès que $i(j) - c(j) < 0$.

Les $(m - i(j) + 1)$ bits de poids les plus forts de d restent inchangés lors de l'étape de randomisation si (condition de consistance) :

$$10 \quad d_{i(j)-1 \rightarrow i(j)-c(j)} \geq d_{m \rightarrow i(j)}$$

$$(i(j) - 1) - (i(j) - c(j)) \geq m - i(j) \Leftrightarrow c(j) \geq m - i(j) + 1$$

Selon une première variante du deuxième mode de réalisation, on choisit $c(j)$ égal à $m - i(j) + 1$. Avec la
 15 condition $i(j) \geq c(j) \geq 0$, la condition $c(j) \geq m - i(j) + 1$ est vérifiée si $2 \cdot i(j) \geq m + 1$.

L'algorithme SAM modifié selon cette variante peut alors s'écrire (algorithme II) :

```

    Entrée :  $x, d = (d_m, \dots, d_0)_2$ 
    Sortie :  $y = x^d \bmod N$ 
5      R0 <- 1 ; R1 <- 1 ; R2 <- x ;
      i <- m ; c <- -1 ;  $\sigma$  <- 1
      tant que  $i \geq 0$ , faire :
          R0 <-  $R0 \times R0 \bmod N$ 
          si  $d_i = 1$  alors R0 <-  $R0 \times R2 \bmod N$  fin si
10      si  $(2i \geq m+1)$  ET  $(\sigma = 1)$  alors c <-  $m-i+1$ 
          sinon  $\sigma = 0$ 

          fin si
           $\rho \leftarrow R\{0, 1\}$ 
           $\varepsilon \leftarrow \rho$  ET  $(d_{i-1} \rightarrow i-c \geq d_{m \rightarrow i})$  ET  $\sigma$ 
15      si  $\varepsilon = 1$  alors
          R1 <- R0 ;  $\sigma \leftarrow 0$ 
           $d_{i-1} \rightarrow i-c \leftarrow d_{i-1} \rightarrow i-c - d_{m \rightarrow i}$ 
          fin si
          si c = 0 alors R0 <-  $R0 \times R1 \bmod N$  ;  $\sigma \leftarrow 1$ 
20      fin si
          c <- c-1 ; i <- i-1
      fin tant que
    retourner R0

```

On notera que l'algorithme I correspond à l'algorithme II dans le cas où $c(j) = i(j)$ pour tout j . On notera également que dans l'algorithme II, la condition de consistance $(d_{i(j)-1 \rightarrow i(j)-c(j)} \geq d_{m \rightarrow i(j)})$ est satisfaite durant la première partie de l'algorithme, en considérant de manière approximative que $d_{i(j)-1 \rightarrow i(j)-c(j)}$ et $d_{m \rightarrow i(j)}$ sont des nombres aléatoires de $(m-i(j)+1)$ bits. On notera, dans cet algorithme, que tous les bits de l'exposant sont randomisés.

Selon une deuxième variante du deuxième mode de réalisation, on choisit $c(j)$ aléatoire et compris entre $i(j)$ et $m-i(j)+1$.

On a vu précédemment que la condition $c(j) \geq m - i(j) + 1$ doit être vérifiée. En posant $c(j) = m - i(j) + 1 + v(j)$, on doit donc vérifier $v(j) \geq 0$. Par ailleurs, avec la condition $i(j) \geq c(j) \geq 0$, il vient $2.i(j) \geq m+1+v(j)$. Donc la plus grande valeur possible pour $v(j)$ est $2.i(j) - m - 1$ et donc, comme $v(j) \geq 0$, le paramètre $c(j) = m - i(j) + 1 + v(j)$ peut prendre toute valeur dans l'ensemble $\{m - i(j) + 1, \dots, i(j)\}$. On peut alors généraliser l'algorithme II en choisissant $c(j)$ aléatoire dans l'ensemble $\{m - i(j) + 1, \dots, i(j)\}$.

Dans l'exemple particulier de l'algorithme II cela revient à remplacer l'instruction :

si $(2i \geq m+1)$ ET $(\sigma = 1)$ alors $c \leftarrow m-i+1$

par l'instruction :

si $(2i \geq m+1)$ ET $(\sigma = 1)$ alors $c \leftarrow R\{m-i+1, i\}$

ce qui donne l'algorithme III suivant :

Entrée : $x, d = (d_m, \dots, d_0)_2$

Sortie : $y = x^d \bmod N$

```

5      R0 <- 1 ; R1 <- 1 ; R2 <- x ;
      i <- m ; c <- -1 ;  $\sigma$  <- 1
      tant que  $i \geq 0$ , faire :
          R0 <-  $R0 \times R0 \bmod N$ 
          si  $d_i = 1$  alors  $R0 \leftarrow R0 \times R2 \bmod N$  fin si
10      si  $(2i \geq m+1)$  ET  $(\sigma = 1)$ 
          alors  $c \leftarrow R\{m-i+1, i\}$ 
          sinon  $\sigma = 0$ 
      fin si
       $\rho \leftarrow R\{0, 1\}$ 
15       $\varepsilon \leftarrow \rho$  ET  $(d_{i-1} \rightarrow i-c \geq d_{m \rightarrow i})$  ET  $\sigma$ 
      si  $\varepsilon = 1$  alors
          R1 <- R0 ;  $\sigma \leftarrow 0$ 
           $d_{i-1} \rightarrow i-c \leftarrow d_{i-1} \rightarrow i-c - d_m \rightarrow i$ 
      fin si
20      si  $c = 0$  alors  $R0 \leftarrow R0 \times R1 \bmod N$  ;  $\sigma \leftarrow 1$ 
      fin si
       $c \leftarrow c-1$  ;  $i \leftarrow i-1$ 
      fin tant que
      retourner R0

```

25 Une valeur plus grande pour $v(j)$ augmente la probabilité de succès pour la condition de consistance (et donc pour le choix d'une randomisation). D'un autre côté, cela réduit également les valeurs possibles de l'indexe i vérifiant la condition $2.i(j) \geq m+1+v(j)$.

30

La fréquence d'occurrence de la valeur $p = 0$ de la variable booléenne p est un paramètre du procédé permettant de choisir le meilleur compromis entre

performance et sécurité, en fonction de l'application envisagée : plus on réalise d'étapes de randomisation, plus on pénalise le temps de calcul global ; inversement, moins on réalise d'étapes de randomisation, plus on
5 facilite les attaques par recherche exhaustive.

Un bon moyen pour minimiser le coût des opérations additionnelles consiste à modifier légèrement le générateur de nombres aléatoires produisant le nombre ρ de sorte que, lorsque le poids de Hamming de $d-z$ (z peut
10 avoir différentes valeurs en fonction de $b(i)$, selon le mode de réalisation envisagé) est plus faible que le poids de Hamming de d , ρ a une plus forte probabilité de valoir 1, et inversement. Avec cette astuce, l'algorithme va tendre à sélectionner le cas ayant le plus faible
15 poids de Hamming, c'est-à-dire la branche la plus rapide.

On note seulement que cet algorithme ne peut pas toujours sélectionner la branche la plus rapide, sinon, il deviendrait déterministe et donc facilement attaquable.

20 Selon un troisième mode de réalisation de l'invention, on choisit au début du procédé un nombre u aléatoire de v bits et on mémorise x^u dans le registre $R1$. De préférence, le nombre u est modifié plusieurs fois au cours du procédé, pour augmenter le facteur aléatoire
25 dans le procédé.

Puis, au cours du calcul, pour un rang $i(j)$ donné, on se demande, pour un paquet w de v bits de d tel que $w > u$, si le calcul de x^w est plus coûteux (en terme de temps de calcul) que celui de $x^{(w-u)} * x^u$.

Pour répondre à cette question, il suffit de déterminer si $H(w) > H(w-u) + 1$. $H(w)$ est le poids de Hamming de w , il est représentatif du coût de l'opération x^w . $H(w-u)$ est le poids de Hamming de $x^{(w-u)}$, représentatif de $x^{(w-u)}$. Le terme " $+ 1$ " est représentatif du coût de la multiplication de $x^{(w-u)}$ par x^u (x^u étant mémorisé par ailleurs).

Si le calcul de x^w est moins coûteux que le calcul de $x^{(w-u)} * x^u$, alors on continue le procédé. Sinon, si le calcul de x^w est plus coûteux que le calcul de $x^{(w-u)} * x^u$, alors on remplace le paquet w de bits de d par le nombre $w-u$. L'étape de consolidation (ici, une multiplication par x^u , qui se traduit par l'opération $R0 \leftarrow R0 \times R1 \bmod N$) sera réalisée lorsque tous les bits de d modifiés auront été utilisés.

Par rapport aux deux précédents modes de réalisation décrits, ce troisième mode de réalisation présente l'avantage d'être plus rapide, puisque, pour effectuer une randomisation, on choisit chaque fois le chemin le plus rapide (le moins coûteux). Ainsi, on montre expérimentalement que la complexité de ce procédé est d'environ 1.4. La complexité est le nombre moyen de multiplications de contenus de registres réalisées pour chaque bit de l'exposant d . La complexité d'un algorithme SAM non protégé est de 1.5 ; la complexité des procédés selon le premier ou le deuxième modes de réalisation de l'invention est quant à elle légèrement supérieure à 1.5.

Par ailleurs, dans ce troisième mode de réalisation, la source d'aléatoire (le nombre u) est extérieure au procédé. Enfin, les ressources (notamment le nombre de registres) utilisées sont les mêmes.

Ce troisième mode de réalisation peut être concrétisé par l'algorithme IV suivant :

```

Entrée :  $x, d = (d_m, \dots, d_0)_2$ 
5 Paramètres :  $v, k$ 
Sortie :  $y = x^d \bmod N$ 
   $R0 \leftarrow 1 ; R2 \leftarrow x ; i \leftarrow m ; L = \{\}$ 
  tant que  $i \geq 0$ , faire
     $R0 \leftarrow R0 \times R0 \bmod N$ 
10   si  $d_i = 1$  alors  $R0 \leftarrow R0 \times R2 \bmod N$  fin si
    si  $i = m \bmod ((m+1)/k)$  alors  $\sigma \leftarrow 1$  fin si
    si  $\sigma = 1$  et  $L = \{\}$  alors
      (modification du nombre u en cours de procédé)
       $\sigma \leftarrow 0 ; u \leftarrow R\{0, \dots, 2^v-1\} ;$ 
15      $R1 = x^u \bmod N$ 
      fin si
       $w \leftarrow d_{i \rightarrow i-v+1}$ 
       $h \leftarrow H(w)$ 
      si  $w \geq u$  alors  $\Delta \leftarrow w - u ; h_\Delta \leftarrow 1 + H(\Delta)$ 
20       sinon  $h_\Delta \leftarrow v+2$ 
      fin si
       $\rho \leftarrow R\{0, 1\}$ 
      si  $[(\sigma=0) \wedge (i-v+1 \geq 0)] \wedge$ 
         $[(h > h_\Delta) \text{ OU } ((\rho=1) \text{ ET } (h=h_\Delta))]$  alors
25        (on choisit de réaliser  $x^{(w-u)}$ )
         $d_{i \rightarrow i-v+1} \leftarrow \Delta ; L \leftarrow L \cup \{i-v+1\}$ 
      fin si
      si  $(i \in L)$  alors
         $R0 \leftarrow R0 \times R1 \bmod N$ 
30         $L \leftarrow L \setminus \{i\}$ 
      fin si
       $i \leftarrow i-1$ 
  fin tant que
  retourner  $R0$ 

```

Dans cet exemple, l'ensemble L contient la liste des indices pour lesquels une étape de consolidation doit être réalisée. L'instruction "si $d_i = 1$ alors $R0 \leftarrow R0 \times R2 \bmod N$ fin si" est l'instruction classique d'un algorithme SAM, réalisée pour chaque valeur de i .

L'exposant d est ici découpé en k blocs, de taille identique si $m+1$ est divisible par k ou de taille identique à une unité près sinon.

A chaque début de bloc (c'est à dire pour $i = m \bmod ((l+1)/k)$), on met la variable σ à 1. Ensuite, quand σ vaut 1, il faut attendre que l'ensemble L soit vide avant d'effectuer une nouvelle étape de randomisation. A chaque étape de consolidation, on enlève un indice $i(j)$ correspondant de l'ensemble L (instruction $L \leftarrow L \setminus \{i\}$). Lorsque l'ensemble L est vidé, une nouvelle valeur de u peut être choisie et x^u est calculé par un algorithme SAM classique utilisant les registres $R1$ et $R2$.

Au milieu de chaque bloc ($\sigma = 0$), on réalise une ou plusieurs étapes de randomisation, lorsque $h > h_\Delta$ ou ($p=1$ et $h = h_\Delta$), et on mémorise à chaque fois (instruction $L \leftarrow L \cup \{i-v+1\}$) l'indice $i(j)-v+1$ auquel on devra effectuer une étape de consolidation. Il faut donc que $i-v+1$ soit un indice de consolidation valable, c'est-à-dire que $i-v+1 \geq 0$ (condition de consistance). A chaque étape de randomisation, si $h > h_\Delta$, on choisit de réaliser l'opération $x^{(w-u)} \times x^u$, moins coûteuse, et on modifie les bits de d en conséquence ($d_{i \rightarrow i-v+1} \leftarrow \Delta$). Si $h < h_\Delta$, on choisit de réaliser x^w , moins coûteuse, et on ne modifie pas d . Si $h = h_\Delta$, on choisit aléatoirement ($p = 0$ ou 1 aléatoire) de réaliser $x^{(w-u)} \times x^u$ ou x^w .

A titre indicatif, on donne le nombre moyen de multiplications modulaires nécessaires pour réaliser une exponentiation de longueur 1024 avec l'algorithme SAM, protégé ou non :

- 5 • SAM sans protection : 1536 multiplication
 - SAM protégé en ajoutant un multiple de $\Phi(n)$ ($r \cdot \Phi(n)$ avec r de 64 bits) ajouté à l'exposant d (art antérieur) : 1536 + 96 multiplications
 - 10 • SAM protégé selon l'algorithme II ou III : 1536 + 10 multiplications
 - SAM protégé selon l'algorithme I' : 1536+512 multiplications. $\bar{\rho}$, $\bar{\rho}$ étant la valeur moyenne de ρ
 - SAM protégé selon l'algorithme IV : 1443 multiplications
- 15 On constate par ces exemples qu'un algorithme protégé selon l'invention est très efficace, en terme de multiplications réalisées (et donc de temps de calcul).

REVENDEICATIONS

1. Procédé cryptographique dans un composant électronique au cours duquel on réalise une exponentiation modulaire de type x^d , avec d un exposant entier de $m+1$ bits, en balayant les bits de d de gauche à droite dans une boucle indicée par i variant de m à 0 et en calculant et en mémorisant dans un accumulateur (R0), à chaque tour de rang i , un résultat partiel actualisé égal à $x^{b(i)}$, $b(i)$ étant les $m-i+1$ bits de poids les plus forts de l'exposant d ($b(i) = d_{m \rightarrow i}$),

10 le procédé étant caractérisé en ce que, à la fin d'un tour de rang $i(j)$ ($i = i(0)$) choisi aléatoirement, on réalise une étape E1 de randomisation au cours de laquelle :

15 E1 : on soustrait un nombre z ($z = b(i(j))$, $z = b(i(j)).2^r$, $z = u$) aléatoire à une partie des bits de d non encore utilisés ($d_{i-1 \rightarrow 0}$) dans le procédé

puis, après avoir utilisé les bits de d modifiés par l'étape de randomisation E1, on réalise une étape de consolidation E2 au cours de laquelle :

20 E2 : on mémorise ($R0 \leftarrow R1 \times R0$) dans l'accumulateur (R0) le résultat de la multiplication du contenu de l'accumulateur ($x^{b(i)}$) par un nombre fonction de x^z mémorisé dans un registre (R1).

2. Procédé selon la revendication précédente, dans lequel l'étape E1 est répétée une ou plusieurs fois, à la fin de différents tours de rang $i(j)$ ($i = i(0)$, $i = i(1)$, ...) choisis aléatoirement entre 0 et m .

REVENDEICATIONS

1. Procédé cryptographique dans un composant électronique au cours duquel on réalise une exponentiation modulaire de type x^d , avec d un exposant entier de $m+1$ bits, en balayant les bits de d de gauche à droite dans une boucle indicée par i variant de m à 0 et en calculant et en mémorisant dans un accumulateur (R0), à chaque tour de rang i , un résultat partiel actualisé égal à $x^{b(i)}$, $b(i)$ étant les $m-i+1$ bits de poids les plus forts de l'exposant d ($b(i) = d_{m \rightarrow i}$),
 - 10 le procédé étant caractérisé en ce que, à la fin d'un tour de rang $i(j)$ ($i = i(0)$) choisi aléatoirement, on réalise une étape E1 de randomisation au cours de laquelle :
 - 15 E1 : on soustrait un nombre z ($z = b(i(j))$, $z = b(i(j)).2^v$, $z = u$) aléatoire à une partie des bits de d non encore utilisés ($d_{i-1 \rightarrow 0}$) dans le procédé
 - puis, après avoir utilisé les bits de d modifiés par l'étape de randomisation E1, on réalise une étape de consolidation E2 au cours de laquelle :
 - 20 E2 : on mémorise ($R0 \leftarrow R1 \times R0$) dans l'accumulateur (R0) le résultat de la multiplication du contenu de l'accumulateur ($x^{b(i)}$) par un nombre fonction de x^z mémorisé dans un registre (R1).
2. Procédé selon la revendication précédente, dans lequel l'étape E1 est répétée une ou plusieurs fois, à la fin de différents tours de rang $i(j)$ ($i = i(0)$, $i = i(1)$, ...) choisis aléatoirement entre 0 et m .

3. Procédé selon la revendication précédente, dans lequel, à chaque tour i , on décide aléatoirement ($p=1$) si on réalise l'étape E1 ou pas.

5 4. Procédé cryptographique selon l'une des revendications 1 à 3, dans lequel le nombre z ($z = b(i(j))$, $z = b(i(j)).2^v$) est fonction de l'exposant d , dans lequel, lors de l'étape de randomisation, on mémorise ($R1 \leftarrow R0 \times R1$) également dans un registre (R1) le résultat de la multiplication du contenu de
10 l'accumulateur ($x^{b(i)}$) par le contenu du dit registre (R1).

5. Procédé selon la revendication 4, dans lequel l'étape de consolidation E2 est réalisée après le dernier tour de rang i égal à 0.

15 6. Procédé selon la revendication précédente, au cours duquel, lors de l'étape E1, on soustrait à d le nombre $b(i)$.

3. Procédé selon la revendication précédente, dans lequel, à chaque tour i , on décide aléatoirement ($p=1$) si on réalise l'étape E1 ou pas.
4. Procédé cryptographique selon l'une des revendications 1 à 3, dans lequel le nombre z ($z = b(i(j))$, $z = b(i(j)).2^v$) est fonction de l'exposant d , dans lequel, lors de l'étape de randomisation, on mémorise ($R1 \leftarrow R0 \times R1$) également dans un registre (R1) le résultat de la multiplication du contenu de l'accumulateur ($x^{b(i)}$) par le contenu du dit registre (R1).
5. Procédé selon la revendication 4, dans lequel l'étape de consolidation E2 est réalisée après le dernier tour de rang i égal à 0.
6. Procédé selon la revendication précédente, au cours duquel, lors de l'étape E1, on soustrait à d le nombre $b(i)$.
7. Procédé selon la revendication 6, au cours duquel, pour calculer $y = x^d \bmod N$, on réalise les étapes suivantes :
- A : initialisation des registres R0, R1, R2 et d'une variable de boucle i :
- $R0 \leftarrow 1$; $R1 \leftarrow 1$; $R2 \leftarrow x$; $i \leftarrow m$
- B : tant que $i \geq 0$, répéter les étapes suivantes :
- a : multiplication du contenu du registre R0 par lui-même modulo N et mémorisation du résultat dans le registre R0
- $R0 \leftarrow R0 \times R0 \bmod N$
- b : si $d_i = 1$ alors multiplication du contenu du registre R0 par le contenu du registre R2 modulo N et mémorisation du résultat dans R0
- si $d_i = 1$ alors $R0 \leftarrow R0 \times R2 \bmod N$
- c : choix aléatoire d'une valeur de la variable p parmi deux valeurs, 0 et 1
- $p \leftarrow R\{0, 1\}$

7. Procédé selon la revendication 6, au cours duquel on réalise :

Entrée : x , $d = (d_m, \dots, d_0)_2$

Sortie : $y = x^d \bmod N$

```

5      R0 <- 1 ; R1 <- 1 ; R2 <- x ; i <- m
      tant que  $i \geq 0$ , faire :
          R0 <-  $R0 \times R0 \bmod N$ 
          si  $d_i = 1$  alors  $R0 \leftarrow R0 \times R2 \bmod N$ 
           $\rho \leftarrow R\{0, 1\}$ 
10      si  $((\rho = 1) \text{ ET } d_{i-1 \rightarrow 0} \geq d_{m-i})$  alors
           $d \leftarrow d - d_{m \rightarrow i}$ 
           $R1 \leftarrow R1 \times R0 \bmod N$ 
          fin si
           $i \leftarrow i - 1$ 
15      fin tant que
           $R0 \leftarrow R0 \times R1 \bmod N$ 
      retourner R0

```

8. Procédé selon la revendication 5, au cours duquel l'étape E1 est modifiée comme suit :

```

20      E1 : on soustrait à  $d$  un nombre égal à  $g.b(i)$ ,  $g$ 
          étant un nombre entier positif ; on élève le résultat
          partiel actuel ( $x^{b(i)}$ ) à la puissance  $g$  et on
          mémorise le résultat dans le registre (R1).

```

```

25      9. Procédé selon la revendication précédente, dans
          lequel  $g$  est égal à  $2^\tau$ ,  $\tau$  étant un nombre aléatoire choisi
          entre 0 et T.

```

```

d :      si ((p = 1) ET  $d_{i-1 \rightarrow 0} \geq d_{m-i}$  alors
          aa : soustraction des bits de rangs i à m de la variable d à
              la variable d et mémorisation dans la variable d
               $d \leftarrow d - d_{m \rightarrow i}$ 
5          bb : multiplication du contenu du registre R1 par le contenu
              du registre R0 modulo N et mémorisation du résultat dans R1
               $R1 \leftarrow R1 \times R0 \bmod N$ 
              fin si
          e : décrémentation de la variable i :
10           $i \leftarrow i - 1$ 
              fin tant que
          C : multiplication du contenu du registre R0 par le contenu du registre
              R1 modulo N et mémorisation du résultat dans R0
               $R0 \leftarrow R0 \times R1 \bmod N$ 
15      D :      retourner R0

```

8. Procédé selon la revendication 5, au cours duquel l'étape E1 est modifiée comme suit :

```

E1 : on soustrait à d un nombre égal à  $g.b(i)$ , g
    étant un nombre entier positif ; on élève le résultat
20 partiel actuel ( $x^{b(i)}$ ) à la puissance g et on
    mémorise le résultat dans le registre (R1).

```

9. Procédé selon la revendication précédente, dans lequel g est égal à 2^τ , τ étant un nombre aléatoire choisi entre 0 et T.

25 10. Procédé selon la revendication précédente, dans lequel, pour calculer $y = x^d \bmod N$, on réalise :

```

A : initialisation des registres R0, R1, R2 et d'une variable de
    boucle i :
     $R0 \leftarrow 1 ; R1 \leftarrow 1 ; R2 \leftarrow x ; i \leftarrow m$ 
30      B :      tant que  $i \geq 0$ , faire :
          a : multiplication du contenu du registre R0 par lui-même modulo N
              et mémorisation du résultat dans le registre R0
               $R0 \leftarrow R0 \times R0 \bmod N$ 
          b : si  $d_i = 1$  alors multiplication du contenu du registre R0 par
35 le contenu du registre R2 modulo N et mémorisation du résultat
              dans R0
              si  $d_i = 1$  alors  $R0 \leftarrow R0 \times R2 \bmod N$ 

```

10. Procédé selon la revendication précédente, dans lequel on réalise :

Entrée : x , $d = (d_m, \dots, d_0)_2$

Sortie : $y = x^d \bmod N$

```

5      R0 <- 1 ; R1 <- 1 ; R2 <- x ; i <- m
      tant que i ≥ 0, faire :
          R0 <- R0×R0 mod N
          si  $d_i = 1$  alors R0 <- R0×R2 mod N
           $\rho \leftarrow R\{0, 1\}$  ;  $\tau \leftarrow R\{0, \dots, T\}$ 
10      si (( $\rho = 1$ ) ET ( $d_{i-1 \rightarrow \tau} \geq d_{m-i}$ )) alors
           $d_{i-1 \rightarrow \tau} \leftarrow d_{i-1 \rightarrow \tau} - d_{m-i}$ 
          R3 <- R0
          tant que ( $\tau > 0$ ) faire
              R3 <- R32 mod N ;  $\tau \leftarrow \tau - 1$ 
15      fin tant que
          R1 <- R1×R3 mod N
      fin si
      i <- i-1
      fin tant que
20      R0 <- R0×R1 mod N
      retourner R0

```

11. Procédé selon l'une des revendications 1 à 4, dans lequel l'étape de consolidation E2 est réalisée à la fin du rang utilisant le dernier bit de d modifié lors de l'étape E1.

12. Procédé selon la revendication 11, au cours duquel, lors de l'étape E1, on soustrait le nombre $b(i)$ aux bits de d de rang $i(j) - c(j)$ à $i(j) - 1$, $c(j)$ étant un nombre entier et on mémorise le contenu de l'accumulateur ($x^{b(i(j))}$) dans le registre (R1).

13. Procédé selon la revendication précédente, au cours duquel, lors du tour de rang $i(j+1)$, on choisit

```

c : choix aléatoire d'une valeur de la variable  $p$  parmi deux
valeurs, 0 et 1 :
     $p \leftarrow R\{0, 1\}$  ;

d : choix aléatoire d'une valeur de la variable  $\tau$  parmi les
5 valeurs 0, ..., T :
     $\tau \leftarrow R\{0, \dots, T\}$ 
e :    si  $((p = 1) \text{ ET } (d_{i-1 \rightarrow \tau} \geq d_{m-i}))$  alors
        aa : soustraction des bits de rangs  $i$  à  $m$  de la variable  $d$ 
            aux bits de rangs  $\tau$  à  $i-1$  de la variable  $d$  et mémorisation
10 dans les bits de rangs  $\tau$  à  $i-1$  de la variable la variable  $d$ 
                 $d_{i-1 \rightarrow \tau} \leftarrow d_{i-1 \rightarrow \tau} - d_{m \rightarrow i}$ 
        bb : mémorisation du registre R0 dans le registre R3 :
                 $R3 \leftarrow R0$ 
        cc : tant que  $(\tau > 0)$ , élever au carré modulo  $N$  le contenu du
15 registre R3, mémoriser le résultat dans R3 et décrémenter la
            variable  $\tau$ 
                tant que  $(\tau > 0)$  faire
                     $R3 \leftarrow R3^2 \bmod N$  ;  $\tau \leftarrow \tau - 1$ 
                fin tant que
        dd : multiplication du contenu du registre R1 par le contenu
20 du registre R3 modulo  $N$  et mémorisation du résultat dans le
            registre R1
                 $R1 \leftarrow R1 \times R3 \bmod N$ 
            fin si
        f : décrémenter la variable  $i$ 
25              $i \leftarrow i - 1$ 
            fin tant que
C : multiplication du contenu du registre R0 par le contenu du registre
R1 modulo  $N$  et mémorisation du résultat dans le registre R0
30      $R0 \leftarrow R0 \times R1 \bmod N$ 
D :    retourner R0

```

11. Procédé selon l'une des revendications 1 à 4, dans lequel l'étape de consolidation E2 est réalisée à la fin du rang utilisant le dernier bit de d modifié lors de
35 l'étape E1.

12. Procédé selon la revendication 11, au cours duquel, lors de l'étape E1, on soustrait le nombre $b(i)$ aux bits de d de rang $i(j) - c(j)$ à $i(j)-1$, $c(j)$ étant un nombre

aléatoirement de réaliser l'étape E1 uniquement si $i(j+1) \leq i(j) - c(j)$. ($\sigma = 1$ sémaphore libre).

14. Procédé selon la revendication 12 ou 13, dans lequel $c(j)$ est égal à $m - i(j) + 1$.

5 15. Procédé selon la revendication précédente, au cours duquel on réalise les étapes suivantes :

Entrée : x , $d = (d_m, \dots, d_0)_2$

Sortie : $y = x^d \bmod N$

$R0 \leftarrow 1$; $R1 \leftarrow 1$; $R2 \leftarrow x$;

10 $i \leftarrow m$; $c \leftarrow -1$; $\sigma \leftarrow 1$

tant que $i \geq 0$, faire :

$R0 \leftarrow R0 \times R0 \bmod N$

si $d_i = 1$ alors $R0 \leftarrow R0 \times R2 \bmod N$ fin si

si $(2i \geq m+1)$ ET ($\sigma = 1$) alors $c \leftarrow m - i + 1$

15 sinon $\sigma = 0$

fin si

$p \leftarrow R\{0, 1\}$

$\varepsilon \leftarrow p$ ET $(d_{i-1} \rightarrow i - c \geq d_{m \rightarrow i})$ ET σ

si $\varepsilon = 1$ alors

20 $R1 \leftarrow R0$; $\sigma \leftarrow 0$

$d_{i-1} \rightarrow i - c \leftarrow d_{i-1} \rightarrow i - c - d_m \rightarrow i$

fin si

si $c = 0$ alors

$R0 \leftarrow R0 \times R1 \bmod N$; $\sigma \leftarrow 1$

25 fin si

$c \leftarrow c - 1$; $i \leftarrow i - 1$

fin tant que

retourner $R0$

30 16. Procédé selon la revendication 12 ou 13, dans lequel $c(j)$ est choisi aléatoirement entre $i(j)$ et $m - i(j) + 1$.

entier et on mémorise le contenu de l'accumulateur $(x^b(i(j)))$ dans le registre (R1).

13. Procédé selon la revendication précédente, au cours duquel, lors du tour de rang $i(j+1)$, on choisit aléatoirement de réaliser l'étape E1 uniquement si $i(j+1) \leq i(j)-c(j)$. ($\sigma = 1$ sémaphore libre).

14. Procédé selon la revendication 12 ou 13, dans lequel $c(j)$ est égal à $m-i(j)+1$.

15. Procédé selon la revendication précédente, au cours duquel, pour calculer $y = x^d \bmod N$, on réalise les étapes suivantes :

A : initialisation des registres R0, R1, R2 et des variables i, c, σ :

R0 \leftarrow 1 ; R1 \leftarrow 1 ; R2 \leftarrow x ;

i \leftarrow m ; c \leftarrow -1 ; $\sigma \leftarrow$ 1

15 B : tant que i \geq 0, faire :

a : multiplication du contenu du registre R0 par lui-même modulo N et mémorisation du résultat dans le registre R0

R0 \leftarrow R0 \times R0 mod N

20 b : si $d_i = 1$ alors multiplication du contenu du registre R2 par le contenu du registre R0 modulo N et mémorisation du résultat dans le registre R0

si $d_i = 1$ alors R0 \leftarrow R0 \times R2 mod N fin si

c : si ($2i \geq m+1$) ET ($\sigma = 1$) alors mise à jour de la variable c, sinon mise à zéro de la variable σ

25 si ($2i \geq m+1$) ET ($\sigma = 1$) alors c \leftarrow m-i+1
sinon $\sigma = 0$

fin si

d : choix aléatoire d'une valeur de la variable p parmi deux valeurs, 0 et 1 :

30 p \leftarrow R{0, 1}

e : mise à jour de la variable ε

$\varepsilon \leftarrow$ p ET ($d_{i-1} \rightarrow i-c \geq d_{m \rightarrow i}$) ET σ

35 f : si $\varepsilon = 1$ alors mémorisation du registre R0 dans le registre R1, mise à zéro de la variable σ , soustraction des bits de rangs i à m de la variable d aux bits de rangs i-c à i-1 de la variable d et mémorisation du résultat dans les bits de rangs i-c à i-1 de la variable d

17. Procédé selon la revendication précédente, au cours duquel on réalise :

Entrée : $x, d = (d_m, \dots, d_0)_2$

Sortie : $y = x^d \bmod N$

```

5      R0 <- 1 ; R1 <- 1 ; R2 <- x ;
      i <- m ; c <- -1 ;  $\sigma$  <- 1
      tant que  $i \geq 0$ , faire :
          R0 <- R0 x R0 mod N
          si  $d_i = 1$  alors R0 <- R0 x R2 mod N
10         si ( $2i \geq m+1$ ) ET ( $\sigma = 1$ )
            alors c <- R{m-i+1, ..., i}
            sinon  $\sigma = 0$ 
           $\varepsilon$  <-  $\rho$  ET ( $d_{i-1} \rightarrow i-c \geq d_{m \rightarrow i}$ ) ET  $\sigma$ 
          si  $\varepsilon = 1$  alors
15             R1 <- R0 ;  $\sigma$  <- 0
              $d_{i-1} \rightarrow i-c \leftarrow d_{i-1} \rightarrow i-c - d_m \rightarrow i$ 
          fin si
          si c = 0 alors
              R0 <- R0 x R1 mod N ;  $\sigma$  <- 1
20          fin si
          c <- c-1 ; i <- i-1
      fin tant que
      retourner R0

```

18. Procédé selon l'une des revendications 1 à 2, dans lequel le nombre z est un nombre u ($z = u$) de v bits choisi aléatoirement et indépendant de l'exposant d .

19. Procédé selon la revendication précédente, dans lequel, au cours de l'étape E1, le nombre u est soustrait à un paquet w de v bits de d .

20. Procédé selon la revendication précédente, au cours duquel :

- si $H(w-u) + 1 < H(w)$, on choisit de réaliser une étape E1 de randomisation,

```

    si  $\varepsilon = 1$  alors
        R1  $\leftarrow$  R0 ;  $\sigma \leftarrow 0$ 
         $d_{i-1} \rightarrow i-c \leftarrow d_{i-1} \rightarrow i-c - d_m \rightarrow i$ 
    fin si
5      g : si c = 0 alors multiplication du contenu du registre R1 par le
        contenu du registre R0 modulo N et mémorisation du résultat dans
        le registre R0 ; mise à 1 de la variable  $\sigma$ 
        si c = 0 alors
            R0  $\leftarrow$  R0  $\times$  R1 mod N ;  $\sigma \leftarrow 1$ 
10      fin si
        h : décrémentation des variables c et i
            c  $\leftarrow$  c-1 ; i  $\leftarrow$  i-1
        fin tant que
    C :    retourner R0

15  16. Procédé selon la revendication 12 ou 13, dans lequel
    c(j) est choisi aléatoirement entre i(j) et m-i(j)+1.

    17. Procédé selon la revendication précédente, au cours
    duquel, pour calculer  $y = x^d \text{ mod } N$ , on réalise :
    A : initialisation des registres R0, R1, R2 et des variables de boucle
    i, c,  $\sigma$  :
20      R0  $\leftarrow$  1 ; R1  $\leftarrow$  1 ; R2  $\leftarrow$  x ;
        i  $\leftarrow$  m ; c  $\leftarrow$  -1 ;  $\sigma \leftarrow$  1
    B :    tant que i  $\geq$  0, faire :
25      a : multiplication du contenu du registre R0 par lui-même modulo N
        et mémorisation du résultat dans le registre R0
            R0  $\leftarrow$  R0  $\times$  R0 mod N
        b : si  $d_i = 1$  alors multiplication du contenu du registre R2 par
        le contenu du registre R0 modulo N et mémorisation du résultat
        dans le registre R0
30      si  $d_i = 1$  alors R0  $\leftarrow$  R0  $\times$  R2 mod N
        c :    si ( $2i \geq m+1$ ) ET ( $\sigma = 1$ ) alors choix d'une valeur de c
        aléatoirement parmi les entiers compris entre m-i+1 et i, sinon
        mise à zéro de la variable  $\sigma$ 
            si ( $2i \geq m+1$ ) ET ( $\sigma = 1$ )
35      alors c  $\leftarrow$  R{m-i+1, ..., i}
            sinon  $\sigma = 0$ 
        d : mise à jour de la variable c
             $\varepsilon \leftarrow \rho$  ET ( $d_{i-1} \rightarrow i-c \geq d_{m-i}$ ) ET  $\sigma$ 

```

- si $H(w-u) + 1 > H(w)$, on choisit de ne pas réaliser d'étape E1,
- si $H(w-1) + 1 = H(w)$, on choisit aléatoirement de réaliser ou non une étape E1 de randomisation.

5 21. Procédé selon la revendication précédente, au cours duquel on réalise :

Entrée : x , $d = (d_m, \dots, d_0)_2$

Paramètres : v , k

Sortie : $y = x^d \bmod N$

```

10   R0 <- 1 ; R2 <- x ; i <- m ; L = {}
      tant que i ≥ 0, faire
          R0 <- R0×R0 mod N
          si di = 1 alors R0 <- R0×R2 mod N fin si
          si i = m mod ((m+1)/k) alors σ <- 1 fin si
15   si σ = 1 et L = {} alors
          s <- 0 ; u <- R{0, ..., 2v-1} ;
          R1 = xu mod N
          fin si
          w <- di-v+1
20   h <- H(w)
          si w ≥ u alors Δ <- w-u ; hΔ <- 1 + H(Δ)
          sinon hΔ <- v+2
          fin si
          ρ <- R{0, 1}
25   si [(σ=0)^(i-v+1≥0)] ∧
          [(h>hΔ) OU ((ρ=1) ET (h=hΔ))] alors
          di-v+1 <- Δ ; L <- L ∪ {i-v+1}
          fin si
          si (i ∈ L) alors
30   R0 <- R0×R1 mod N
          L <- L \ {i}
          fin si
          i <- i-1
      fin tant que
35 retourner R0

```

e : si $\varepsilon = 1$ alors mémorisation de R0 dans R1, mise à zéro de σ , soustraction des bits de rangs i à m de la variable d aux bits de rangs i-c à i-1 de la variable d et mémorisation du résultat dans les bits de rangs i-c à i-1 de la variable d

5 si $\varepsilon = 1$ alors

 R1 \leftarrow R0 ; $\sigma \leftarrow 0$

$d_{i-1} \rightarrow i-c \leftarrow d_{i-1} \rightarrow i-c - d_m \rightarrow i$

 fin si

10 f : si c = 0 alors multiplication du contenu du registre R1 par le contenu du registre R0 modulo N, mémorisation du résultat dans le registre R0, et mise à 1 de la variable σ

 si c = 0 alors

 R0 \leftarrow R0 \times R1 mod N ; $\sigma \leftarrow 1$

 fin si

15 g : décrémentation des variables c, i

 c \leftarrow c-1 ; i \leftarrow i-1

 fin tant que

 C : retourner R0

18. Procédé selon l'une des revendications 1 à 2, dans

20 lequel le nombre z est un nombre u ($z = u$) de v bits choisi aléatoirement et indépendant de l'exposant d.

19. Procédé selon la revendication précédente, dans lequel, au cours de l'étape E1, le nombre u est soustrait à un paquet w de v bits de d.

25 20. Procédé selon la revendication précédente, au cours duquel :

- si $H(w-u) + 1 < H(w)$, on choisit de réaliser une étape E1 de randomisation,
- si $H(w-u) + 1 > H(w)$, on choisit de ne pas réaliser

30 d'étape E1,

- si $H(w-1) + 1 = H(w)$, on choisit aléatoirement de réaliser ou non une étape E1 de randomisation.

21. Procédé selon la revendication précédente, au cours duquel, v, k étant des paramètres et x, d des données

35 d'entrée, pour calculer $y = x^d \bmod N$, on réalise :

```

A :  initialisation des registres R0, R2, de la variable de boucle i et
    de l'ensemble L :
        R0 <- 1 ; R2 <- x ; i <- m ; L = {}

B :  tant que i ≥ 0, faire
5      a : multiplication du contenu du registre R0 par lui-même modulo N
        et mémorisation du résultat dans le registre R0
            R0 <- R0×R0 mod N

        b : si di = 1 alors multiplication du contenu du registre R2 par
10       le contenu du registre R0 modulo N et mémorisation du résultat
            dans le registre R0
            si di = 1 alors R0 <- R0×R2 mod N fin si

        c : si i = m mod ((m+1)/k) alors mise à 1 de la variable σ
            si i = m mod ((m+1)/k) alors σ <- 1 fin si

        d : si σ = 1 et L = {} alors mise à zéro de la variable s, choix
15       aléatoire de la variable u dans l'ensemble {0, ..., 2v-1}, élévation
            de la donnée x à la puissance u modulo N et mémorisation du
            résultat dans le registre R1
            si σ = 1 et L = {} alors
                s <- 0; u <- R{0, ..., 2v-1} ;
20             R1 = xu mod N
            fin si

        e : mémorisation des bits de rangs i-v+1 à i de la variable d dans
            la variable w
            w <- di-v+1→i

25       f : mémorisation du poids de Hamming de la variable w dans la
            variable h
            h <- H(w)

        g : si w ≥ u alors soustraction de u à la variable w et
            mémorisation dans la variable Δ, mémorisation du poids de Hamming
30       de la variable Δ incrémenté de 1 dans la variable hΔ, sinon
            mémorisation de la variable v incrémentée de 2 dans la variable hΔ
            si w ≥ u alors Δ <- w-u ; hΔ <- 1 + H(Δ)
                sinon hΔ <- v+2
            fin si

35       h : choix aléatoire de la variable u dans l'ensemble {0, 1},
            p <- R{0, 1}

        i : si [(σ=0)^(i-v+1)≥0] ∧
            [(h>hΔ)OU((p=1)ET(h=hΔ))] alors mémorisation de
            Δ dans les bits de rangs i-v+1 à i de la variable d, ajout de la
40       valeur i-v+1 à l'ensemble L
            si [(σ=0)^(i-v+1)≥0] ∧
                [(h>hΔ)OU((p=1)ET(h=hΔ))] alors
                    di-v+1→i <- Δ ; L <- L ∪ {i-v+1}

```

5

10

```
C : retourner R0
```



BREVET D'INVENTION

CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

N° 11235*03

DÉPARTEMENT DES BREVETS

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08

Téléphone : 33 (1) 53 04 53 04 Télécopie : 33 (1) 42 94 86 54

DÉSIGNATION D'INVENTEUR(S) Page N° 1../1..

(À fournir dans le cas où les demandeurs et les inventeurs ne sont pas les mêmes personnes)

INV

Cet imprimé est à remplir lisiblement à l'encre noire

DB 113 @ W / 270601

Vos références pour ce dossier (facultatif)		017069
N° D'ENREGISTREMENT NATIONAL		0314959
TITRE DE L'INVENTION (200 caractères ou espaces maximum)		
Procédé cryptographique d'exponentiation modulaire protégé contre les attaques de type DPA.		
LE(S) DEMANDEUR(S) :		
GEMPLUS Avenue du Pic de Bertagne Parc d'activités de Gemenos 13420 GEMENOS FRANCE		
DESIGNE(NT) EN TANT QU'INVENTEUR(S) :		
1	Nom	CHEVALLIER-MAMES
	Prénoms	Benoît
Adresse	Rue	Résidence Le Général 14 Boulevard Ganteaume
	Code postal et ville	13400 AUBAGNE
Société d'appartenance (facultatif)		
2	Nom	
	Prénoms	
Adresse	Rue	
	Code postal et ville	
Société d'appartenance (facultatif)		
3	Nom	
	Prénoms	
Adresse	Rue	
	Code postal et ville	
Société d'appartenance (facultatif)		
S'il y a plus de trois inventeurs, utilisez plusieurs formulaires. Indiquez en haut à droite le N° de la page suivi du nombre de pages.		
DATE ET SIGNATURE(S) DU (DES) DEMANDEUR(S) OU DU MANDATAIRE (Nom et qualité du signataire) Jean-Louis LECLAIRE - 93.4009		
CABINET BALLOT CONSEILS EN PROPRIÉTÉ INDUSTRIELLE 9, rue Claude Chappe Technopôle Metz 2000 57070 METZ		

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

